RG

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/583,097	08/02/1999	Marc Tremblay	004-1391-1	7166

22120 7590 10/06/2004

ZAGORIN O'BRIEN & GRAHAM, L.L.P.
7600B N. CAPITAL OF TEXAS HWY.
SUITE 350
AUSTIN, TX 78731

EXAMINER
HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 10/06/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/583,097	TREMBLAY, MARC	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 July 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *9-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *9-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *07 August 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 9-36 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Extension of Time and Amendment as received on 7/30/2004.

Amendment Non-Compliance

3. Applicant's amendment fails to comply with the revised 37 CFR 1.121, which is required as of July 30, 2003. More specifically, applicant has used "(Previously Presented)" as a status identifier for claim 35. However, since claim 35 is currently amended, the appropriate status identifier would be "(Currently Amended)." The examiner is refraining from sending out a notice of non-compliance because the examiner feels that this was simply an accident, as appropriate identifiers were used elsewhere, but the examiner asks that applicant carefully check the identifiers before submission, as a notice of non-compliance may be sent out in the future. Please see the attached flyer for more details.

Claim Objections

4. Claim 18 is objected to because of the following informalities: Please replace "intragroup" with --intra-group--. Appropriate correction is required.

5. Claim 28 is objected to because of the following informalities: Please rephrase the phrase "and within group resource allocation" as the better placement of "within" would make the claim more clear and easier to understand. Appropriate correction is required.

6. Claim 31 is objected to because of the following informalities: In line 10, replace "checking as between" with --checking between--. Appropriate correction is required.

7. Claim 33 is objected to because of the following informalities: Replace "a assumption" with --an assumption--. Appropriate correction is required.

8. Claim 34 recites the limitation "the computed alternatives" in the last line. There is insufficient antecedent basis for this limitation in the claim.

Withdrawn Rejections

9. Applicant, via amendment, has overcome the rejections of claims 9-36 by Vegesna. Consequently, these rejections have been withdrawn by the examiner. However, upon further consideration, a new ground(s) of rejection is made below.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 9-36 are rejected under 35 U.S.C. 102(b) as being anticipated by Sites et al., U.S. Patent No. 5,193,167 (herein referred to as Sites).

12. Referring to claim 9, Sites has taught a superscalar processor that, for a given instruction instance, performs, over plural execution cycles of the superscalar processor, instruction grouping for dispatch, including both intra-group and inter-group dependency checking, wherein the instruction grouping for dispatch takes the plural execution cycles to complete. See Fig.3 and column 6, lines 22-26, and note that dual issue is possible (issuing a group of two instructions). In addition, see column 10, line 63, to column 11, line 6-8 (especially, column 11, lines 6-8). Note that over multiple cycles (stages S0-S3) plus any stall cycles, dependencies and resource constraints are checked.

13. Referring to claim 10, Sites has taught a processor as described in claim 9. Sites has further taught:

a) grouping logic implementing plural early pipeline stages of the superscalar processor. See Fig.7 and column 10, lines 64-67, and note that the grouping logic is within the first 4 stages of the pipeline (S0 to S3).

b) plural execution units of varying pipeline depth coupled to receive instructions dispatched from the grouping logic. See Fig.4 and Fig.6 and note the integer and floating-point units. Also, see column 8, line 68, to column 9, line 5, and note that the floating-point divide requires multiple cycles.

14. Referring to claim 11, Sites has taught a processor as described in claim 9. Sites has further taught that the instruction grouping identifies successive groups of instructions from an instruction stream for dispatch to respective ones of plural execution units of the superscalar processor. It is inherent that the system will continue grouping and issuing multiple instructions

per cycle if resources are available. Consequently, previous groups, current groups, and successive groups will all be issued.

15. Referring to claim 12, Sites has taught a processor as described in claim 11. Sites has further taught that the superscalar processor dispatches all instructions from a particular one of the successive groups before dispatching any instructions from a subsequent one of the successive groups. See column 6, lines 26-38 and note that a remaining instruction from a group will be issued before an instruction of another group.

16. Referring to claim 13, Sites has taught a processor as described in claim 9. Sites has further taught that the intra-group dependency checking spans at least two of the plural execution cycles. See column 10, lines 35-41. Note that two cycles are required to fetch multiple instructions and check to see if they can be issued together (stages S0 and S1). It should be realized that fetching is considered a portion of intra-group dependency checking because if the instructions aren't fetched, then they cannot be checked for dependencies. Therefore, fetching is part of the dependency-checking process.

17. Referring to claim 14, Sites has taught a processor as described in claim 9. Sites has further taught that the intra-group dependency checking is independent of the inter-group dependency checking. It is inherent that these two checks are independent, in a sense, because they are two different checks. For instance, intra-group checks must be done to see if the instructions can be issued together. This is dependent on the types of instructions to be issued (some types cannot be issued together). See column 9, lines 46-63. On the other hand, inter-group dependency checking is based on the availability of resources (such as functional units). See column 6, lines 22-29, and note that it is implied that the instructions can issue in parallel,

but the resources are not available for one instruction or the other (determined via inter-group checking).

18. Referring to claim 15, Sites has taught a processor as described in claim 9. Sites has further taught that the data dependency and resource allocation checks in earlier pipeline stages of instruction grouping are based, at least in part, on a predicted subsequent state of the superscalar processor. See column 9, lines 50-57, and column 6, lines 42-44, and note that branches may be issued in parallel and branch prediction is utilized. Consequently, all of the dependency and resource checks for instructions in the predicted path of the branch are based on a predicted subsequent state, which is a state subsequent to the state of the system before the prediction occurs.

19. Referring to claim 16, Sites has taught a processor as described in claim 9. Sites has further taught that non-deterministic conditions are evaluated in a final stage of instruction grouping prior to dispatch. See column 9, lines 61-63. *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." One such condition that is checked would be inter-group dependencies. If an inter-group dependency exists for the two instructions ready to issue, then those instructions will not issue. On the other hand, if an inter-group dependency does not exist for the two instructions, then both of the instructions will be issued simultaneously as a group. According to the definition above, a dependency will exist or not exist, and consequently, the dependency check has more than one result. Therefore, non-deterministic conditions are evaluated.

20. Referring to claim 17, Tremblay has taught a processor comprising:

a) plural functional units that execute instructions in respective numbers of processor cycles. See Fig.4 and Fig.6.

b) grouping logic coupled to the functional units and pipelined to compute, over plural cycles, T, of the processor, a future state, S(t + T), of the processor based on a prior state, S(t), of the processor and based thereon to select a group of instructions from a program sequence thereof for dispatch to the functional units, wherein the future state computing takes the plural cycles to complete. See Fig.3 and column 6, lines 22-26, and note that dual issue is possible (issuing a group of two instructions). In addition, see column 10, line 63, to column 11, line 6-8 (especially, column 11, lines 6-8). Note that multiple instructions may be stalled for multiple cycles due to resource constraints and other reasons (dependencies). It should be realized that S(t) is the state in which two instructions are fetched. From stages S0-S3 of the pipeline (Fig.7 and column 10, lines 64-67), it is determined whether these instruction are able to issue together (intra-group checks) and if resources are available for these instructions (inter-group checks). Consequently, in 4 cycles (stages S0-S3 and assuming no stalling), state S(t + T) will finally be computed. At this point, the group may be issued to the functional units. Therefore, the future state is calculated over 4 cycles.

21. Referring to claim 18, Sites has taught a processor as described in claim 17. Furthermore, claim 18 is rejected for the same reasons set forth in claim 13.

22. Referring to claim 19, Sites has taught a processor as described in claim 17. Furthermore, claim 19 is rejected for the same reasons set forth in claim 14.

23. Referring to claim 20, Sites has taught a processor as described in claim 17. Sites has further taught:

a) intra-group dependencies are checked by the pipelined grouping logic beginning in a first of the T cycles. See column 9, lines 58-63, and note that if the actual instruction within the group conflict with each other, then they will not be issued together. This appears to happen before the inter-group checks as there is no point in determining if resources for both instructions are available if the instructions cannot issue together anyway.

b) non-deterministic dependency conditions are checked during a last of the T cycles. See column 9, lines 58-63. *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." One such condition that is checked would be inter-group dependencies. If an inter-group dependency exists for the two instructions ready to issue, then those instructions will not issue. On the other hand, if an inter-group dependency does not exist for the two instructions, then both of the instructions will be issued simultaneously as a group. According to the definition above, a dependency will exist or not exist, and consequently, the dependency check has more than one result. Therefore, non-deterministic conditions are evaluated. This appears to happen after the intragroup checks because there is no point in determining non-deterministic conditions for both instructions if the instructions cannot issue together anyway. Plus, the passage states that inter-group checks are "also" performed, as in they occur in addition, or after, the intragroup checks.

24. Referring to claim 21, Sites has taught a processor as described in claim 20. Furthermore, claim 21 is rejected for the same reasons set forth in claim 14.

25. Referring to claim 22, Sites has taught a processor as described in claim 17. Sites has further taught that the grouping logic implements T stages of a pipeline of the processor. See Fig.7 and column 10, lines 64-67, and note that the grouping logic is within the first 4 stages of the pipeline (S0 to S3).

26. Referring to claim 23, Sites has taught a processor as described in claim 17. Sites has further taught that T is four. See Fig.7 and column 10, lines 64-67, and note that the grouping logic is within the first 4 stages of the pipeline (S0 to S3). Note the four stages S0-S3 in the grouping process.

27. Referring to claim 24, Sites has taught a processor as described in claim 17. Sites has further taught that the functional units include at least one functional unit capable of receiving and completing an instruction for each of the processor cycles. See column 9, line 67, to column 10, line 2, and column 2, lines 15-18.

28. Referring to claim 25, Sites has taught a processor as described in claim 17. Sites has further taught that the functional units include at least one functional unit requiring multiple of the processor cycles for receiving and completing an instruction. See column 9, lines 3-5.

29. Referring to claim 26, Sites has taught a method of operating a processor, the method comprising:

a) identifying successive groups of instructions for dispatch to respective ones of plural execution units of the processor. See column 6, lines 18-26. Clearly, successive groups of instructions will continue to be issued to functional units shown in Fig.4 and Fig.6.

b) performing, during plural pipelined execution cycles of the processor, dependency checking amongst instructions of a later one of the groups and between the instructions of the later group

and instructions of a preceding one of the groups. See column 6, lines 22-26, and note that a resource check is made. Clearly, if an instruction is already using a floating-point divide unit, for instance, then the successive instruction will have to wait until that unit is available. Also, see Fig.7 and column 10, line 63, to column 11, line 8. Note that over multiple cycles (stages S0-S3) plus any stall cycles, dependencies and resources are checked.

c) dispatching instructions of the later group only after all instructions of the preceding group have been dispatched. See column 6, lines 26-38 and note that a remaining instruction from a group will be issued before an instruction of another group.

d) wherein the performed dependency checking takes the plural pipelined execution cycles to complete. Again, note that over multiple cycles (stages S0-S3) plus any stall cycles, dependencies and resources are checked.

30. Referring to claim 27, Sites has taught a method of operating a processor as described in claim 26. Sites has further taught that the dependency checking is performed by pipelined grouping logic. From stages S0-S3 of the pipeline (Fig.7 and column 10, lines 64-67), instructions are fetched and it is determined whether these instructions are able to issue together based on instruction types and resources used by these instructions (intra-group checks) and if resources are available for these instructions (inter-group checks). Therefore, stages S0-S3 of the pipeline make up the pipelined grouping logic.

31. Referring to claim 28, Sites has taught a method of operating a processor as described in claim 26. Sites has further taught that intra-group dependency checking and within group resource allocation are performed in successive processor cycles by pipelined grouping logic. See column 6, lines 26-29, and note that even if the instructions may issue together (passes intra-

group checks), they still may be unable to issue together if resources are only available for one instruction (determined by inter-group checks). If this is the case, resources are only allocated to the one instruction of the two. Therefore, resource allocation happens only when the instructions are issued and to be issued, the intra-group checks must first be made.

32. Referring to claim 29, Sites has taught a processor as described in claim 26. Furthermore, the method of claim 29 is performed by the processor of claim 14. Consequently, claim 29 is rejected for the same reasons set forth in claim 14.

33. Referring to claim 30, Sites has taught a method of operating a processor as described in claim 26. Sites has further taught that non-deterministic conditions are evaluated in a final one of the pipelined execution cycles implemented by pipelined grouping logic. See column 9, lines 58-63. *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." One such condition that is checked would be inter-group dependencies. If an inter-group dependency exists for the two instructions ready to issue, then those instructions will not issue. On the other hand, if an inter-group dependency does not exist for the two instructions, then both of the instructions will be issued simultaneously as a group. According to the definition above, a dependency will exist or not exist, and consequently, the dependency check has more than one result. Therefore, non-deterministic conditions are evaluated. This appears to happen after the intragroup checks because there is no point in determining non-deterministic conditions for both instructions if the instructions cannot issue together anyway. Plus, the passage states that inter-group checks are "also" performed, as in they occur in addition, or after, the intragroup checks.

34. Referring to claim 31, Sites has taught a method of grouping instructions for dispatch to execution units of a processor, the method comprising:

a) in a first cycle of processor execution, identifying plural candidate instructions for an instruction group. See Fig.3 and note that two instructions are fetched (stage S0 of the pipeline shown in Fig.7) and identified as instructions that may be issued together.

b) in a subsequent cycle of processor execution, beginning intra-group dependency checking as amongst instructions of the instruction group. See column 6, lines 18-21, and note that after the instructions are decoded intra-group checks are made. See column 9, lines 50-57, and note that that instructions must be of the types shown in the table, otherwise a conflict exists between the two instructions and they cannot issue together. Also, it is clear that if both instructions need to use the same unit, then resources will not be available for one of the instructions until the other finishes. Also, the checks will occur after stage S0 in the pipeline (since S0 is the fetch stage). More specifically, the instructions must be fetched before they're checked for intra-group dependencies.

c) in a cycle of processor execution prior to dispatch of any instruction from the instruction group, checking non-deterministic conditions. See column 9, lines 58-63. *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." One such condition that is checked would be inter-group dependencies. If an inter-group dependency exists for the two instructions ready to issue, then those instructions will not issue. On the other hand, if an inter-group dependency does not exist for the two instructions, then both of the instructions will be issued simultaneously

as a group. According to the definition above, a dependency will exist or not exist, and consequently, the dependency check has more than one result. Therefore, non-deterministic conditions are evaluated. This appears to happen after the intragroup checks because there is no point in determining non-deterministic conditions for both instructions if the instructions cannot issue together anyway. Plus, the passage states that inter-group checks are "also" performed, as in they occur in addition, or after, the intragroup checks.

d) in a cycle of processor execution prior to the non-deterministic dependency condition checking, initiating inter-group dependency checking as between instructions of the instruction group and instructions of one or more prior instruction groups. Clearly, before the non-deterministic dependency checking occurs (which is the inter-group checking), the inter-group checking must be initiated. And, it is clear that inter-group checking occurs between groups and that this type of check exists within Sites. That is, if an instruction from group "A" is already using a floating-point divide unit, for instance, then the successive instruction from group "B" will have to wait until that unit is available.

e) wherein, for instruction instances of the instruction group, dependency checking, which includes the intra-group dependency checking and the inter-group dependency checking, takes plural of the cycles of processor execution to complete. From stages S0-S3 of the pipeline (Fig.7 and column 10, lines 64-67), instructions are fetched and it is determined whether these instructions are able to issue together based on instruction types and resources used by these instructions (intra-group checks) and if resources are available for these instructions (inter-group checks). Therefore, multiple cycles are required.

35. Referring to claim 32, Sites has taught a method as described in claim 31. Furthermore, the method of claim 32 is performed by the processor of claim 12. Therefore, claim 32 is rejected for the same reasons set forth in claim 12.

36. Referring to claim 33, Sites has taught a method as described in claim 31. Sites has further taught that the non-deterministic condition checking is performed conservatively, based on an assumption of no change in condition. Clearly, it must be assumed that that there is no change in condition, i.e., dependencies exist (recall that non-deterministic condition checking is synonymous with the inter-group checking in that when performing inter-group checks, a dependency either exists or does not exist). Consequently, dependencies must be checked for.

37. Referring to claim 34, Sites has taught a method as described in claim 31. Sites has further taught:

a) that the non-deterministic condition checking is performed aggressively, computing dispatch conditions for at least two alternatives including no change in condition and condition resolution. Again, the non-deterministic condition checking is synonymous with the inter-group checking in that when performing inter-group checks, a dependency either exists (no change in condition) or does not exist (condition resolution).

b) a control signal is selective for a particular one of the computed alternatives. Clearly, different things happen when a dependency exists (instructions not issued) as opposed to when doesn't exist (instructions are issued). In order to determine which action to take, a control signal is inherently present.

38. Referring to claim 35, Sites has taught an apparatus comprising:

a) plural functional units. See Fig.4 and Fig.6.

b) means for grouping, over plural pipeline stages, instructions for dispatch to respective ones of the functional units, wherein the grouping of a particular set of instruction instances takes plural pipeline stages to complete. From stages S0-S3 of the pipeline (Fig.7 and column 10, lines 64-67), instructions are fetched and it is determined whether these instructions are able to issue together based on instruction types and resources used by these instructions (intra-group checks) and if resources are available for these instructions (inter-group checks). Also, see column 9, lines 45-63, and column 6, lines 18-29. Therefore, multiple cycles are required.

39. Referring to claim 36, Sites has taught an apparatus as described in claim 35. Sites has further taught means for deriving over T processor cycles, a future state S(t + T) based on a present state and means for determining a group of the instructions to dispatch at time t + T. See Fig.3 and column 6, lines 22-26, and note that dual issue is possible (issuing a group of two instructions). In addition, see column 10, line 63, to column 11, line 6-8 (especially, column 11, lines 6-8). Note that multiple instructions may be stalled for multiple cycles due to resource constraints and other reasons (dependencies). It should be realized that S(t) is the state in which two instructions are fetched. From stages S0-S3 of the pipeline (Fig.7 and column 10, lines 64-67), it is determined whether these instruction are able to issue together (intra-group checks) and if resources are available for these instructions (inter-group checks). Consequently, in 4 cycles (stages S0-S3 and assuming no stalling), state S(t + T) will finally be computed. At this point, the group may be issued to the functional units. Therefore, the future state is calculated over 4 cycles.

Response to Arguments

40. Applicant's arguments filed on July 30, 2004, have been fully considered but they are not persuasive. Applicant argues the examiner's definition of "non-deterministic" on page 8 of the remarks.

41. These arguments are not found persuasive for the following reasons:

a) The examiner has provided a valid definition of non-deterministic as claimed. Applicant argues the examiner's definition, yet does not explain what a non-deterministic condition "actually" is. Consequently, the examiner feels that this definition is valid until applicant elaborates, within the claims, on the meaning of non-deterministic, as supported by the specification.

Conclusion

42. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
September 29, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100